Asignatura: Sistemas y señales discretos.
Tema: **La Transformada Rápida de Fourier (FFT)**
Lugar de Ejecución: Instrumentación y control
(Edificio de electrónica)

Objetivos específicos

- Conocer que es la Transformada de Fourier Discreta y sus aplicaciones.
- Resolver problemas que involucren la Transformada de Fourier Discreta.
- Obtener la Transformada Rápida de Fourier de señales reales usando la tarjeta de sonido.

Introducción teórica

6.1 LA TRANSFORMADA RÁPIDA DE FOURIER

El cálculo de la TFD requiere de N^2 multiplicaciones (esto es, $0 \le k < N$, $0 \le n < N$) y el tiempo de computación resulta excesivo cuando N es grande. La clave para métodos computacionales más eficientes es hacer uso de la simetría de los exponenciales complejos tanto como sea posible antes de efectuar las multiplicaciones.
Los avances han producido un tipo de eficientes algoritmos, conocidos como *transformada rápida de Fourier* (TRF), que permite reducir sustancialmente el tiempo de computación. La TRF es un algoritmo (es decir, un método sistemático de efectuar una secuencia de cálculos) que permite calcular la TFD con un mínimo de tiempo de computación. Al margen del propio algoritmo, la interpretación de la TRF es igual a la de la TFD.

El algoritmo que se verá aquí es la formulación de Cooley-Tukey usada comúnmente[1]. Este algoritmo calcula N componentes de frecuencia discretos a partir de N muestras discretas en el tiempo, donde $N = 2^r$, y r es un entero. Esta restricción a una potencia de dos, no es importante en la práctica, siempre que 2^r sea mayor que el número de datos (siempre puede llenarse el resto con ceros de aumento).

Como se vio en la práctica anterior, los parámetros Ω y T no están realmente implicados en el cálculo de la TFD, por lo que se adopta una notación compacta omitiéndolos, y la ecuación (4.2) puede rescribirse como:

$$F_D(n) = \sum_{k=0}^{N-1} f(k) W^{nk}, \qquad (6.1)$$

donde

$$W^l = e^{-j2\pi l/N}, l = 0,1,2,\ldots \qquad (6.2)$$

La ecuación (6.2) describe un fasor de magnitud unitaria y un ángulo de fase de $\theta_l = -2\pi l / N$. Como ejemplo, sea $N = 2^3 = 8$; los valores correspondientes de la ecuación (6.2) están en la figura

[1] J. W. Cooley y J. W. Tukey, "An Algorithm for Machine Calculation of Complex Fourier Series." Math. Comput. 19: 297-301 (Abril 1965).

6.1. En ésta puede verse que W^{l} y W^{-l} están situados simétricamente con respecto al eje real, y rigen las siguientes propiedades de simetría:

$$W^{N} = W^{0} = 1$$
$$W^{N/2} = -1 = -W^{0}$$
$$W^{N-1} = \left[W^{l}\right]^{*}$$

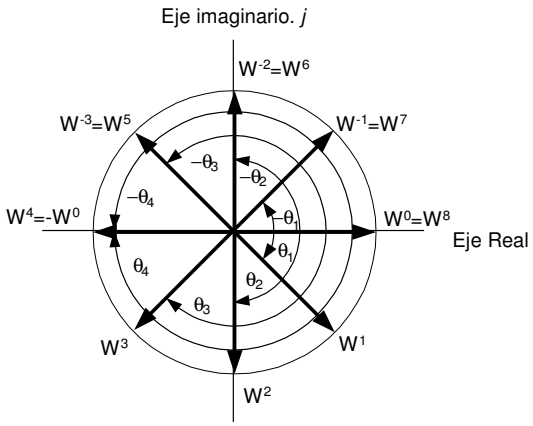


Figura 6.1. Potencias de la función exponencial W para N= 8.

Para aprovechar todas las ventajas del uso de $N = 2^{r}$ se expresan *n* y *k* como números binarios. Supóngase que $r = 2$ (es decir, $N = 4$) y

$$k = (k_1, k_0) = \{00,01,10,11\},$$
$$n = (n_1, n_0) = \{00,01,10,11\}, \qquad (6.4)$$

donde n_0, n_1, k_0, k_1 solo pueden tomar valores de 0 y 1. Una forma compacta de escribir el valor numérico de *k* y *n* es

$$k = 2k_1 + k_0$$
$$n = 2n_1 + n_0 \qquad (6.5)$$

Usando las ecuaciones (6.4) y (6.5) en la (6.1), se tiene

$$F_D(n_1, n_0) = \sum_{k_0=0}^{1}\sum_{k_1=0}^{1} f(k_1, k_0) W^{(2n_1+n_0)(2k_1+k_0)} \qquad (6.6)$$

donde la doble sumatoria es necesaria para indicar la representación binaria completa de *k.* El término exponencial de la ecuación (6.6) es

$$W^{(2n_1+n_0)(2k_1+k_0)} = W^{(2n_1+n_0)2k_1} W^{(2n_1+n_0)k_0}$$
$$= W^{2n_0k_1} W^{(2n_1+n_0)k_0}, \qquad (6.7)$$

porque $W^{4n_1k_1} = 1$ para todos los n_1, k_1 enteros. Este último paso es fundamental para la eficiencia de la TRF porque ahora puede escribirse como

$$F_D(n_1, n_0) = \sum_{k_0=0}^{1} \left[\sum_{k_1=0}^{1} f(k_1, k_0) W^{2n_0k_1} \right] W^{(2n_1+n_0)k_0} \qquad (6.8)$$

Ahora puede mostrarse el algoritmo escribiendo la sumatoria entre corchetes de la ecuación (6.8) como

$$f_1(n_0, k_0) = \sum_{k_1=0}^{1} f(k_1, k_0) W^{2n_0k_1} , \qquad (6.9)$$

y la sumatoria exterior como

$$f_2(n_0, n_1) = \sum_{k_1=0}^{1} f_1(n_0, k_0) W^{(2n_1+n_0)k_0} . \qquad (6.10)$$

De las ecuaciones (6.8) y (6.10) se obtiene

$$F_D(n_1, n_2) = f_2(n_0, n_1) \qquad (6.11)$$

Este último paso se incluye porque el orden de la salida se altera en este algoritmo. Una variante es alterar el orden a la entrada de forma que a la salida sea el correcto. Las ecuaciones (6.9)-(6.11) son las relaciones que comprenden el algoritmo de Cooley-Tukey para $N = 4$. En la figura 6.2 se muestra un gráfico de fluencia de estas relaciones. Para potencias de dos mayores, los algoritmos se procesan de la misma forma, con una sumatoria por cada potencia de dos en la ecuación (6.6).
Para la evaluación directa de la TFD se requieren alrededor de N^2 operaciones complejas de multiplicación y suma, mientras que el algoritmo de la TRF requiere operaciones del orden de $N \log_2 N$. El ahorro neto es apreciable para N grande. Por ejemplo, el tiempo de computación necesario para una TFD de 1024 puntos con evaluación directa es casi 100 veces mayor que si se usa la TRF. Sin embargo, el algoritmo de la TRF requiere considerable espacio de almacenamiento lo que puede limitar su aplicación cuando no se dispone del suficiente espacio.

A continuación se enumeran algunos puntos que resultan útiles al procesar funciones continuas del tiempo con la TRF. Puede verse que muchos de ellos provienen de las consideraciones anteriores sobre la TFD y la transformada de Fourier.

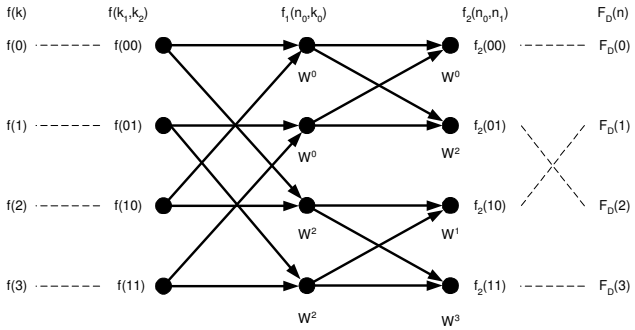


Figura 6.2. Gráfica de flujo de señales del algoritmo Cooley-Tukey para $N = 4$.

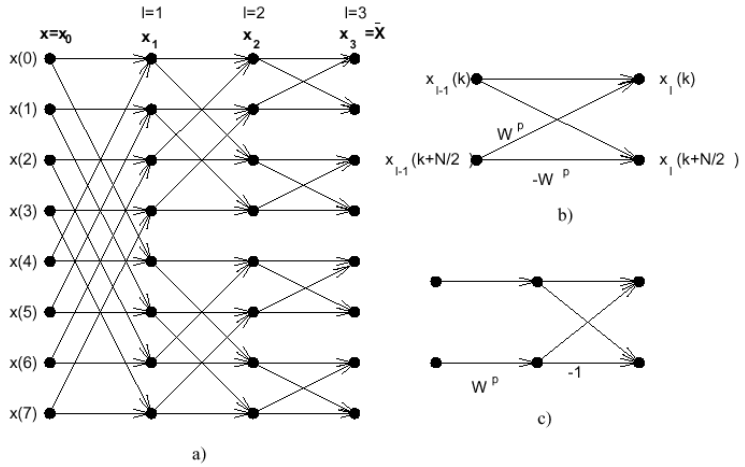
1. El número de muestras N se elige de forma que $N = 2^r$, r entero. Este número puede incluir ceros de aumento [ver (7) más adelante].
2. Para N muestras temporales hay n frecuencias discretas.

3. Como resultado de la extensión periódica, los puntos de muestra 0 y *N* son idénticos en ambos dominios.
4. Los componentes de frecuencia positiva son los que están en (0, *N*/2); los de frecuencia negativa, los que están en (*N*/2, *N*). Esta simetría puede usarse en muestras temporales en tiempo negativo y positivo.
5. Para funciones con valor real, los componentes de frecuencia positiva son los conjugados complejos de los de frecuencia negativa. Los puntos n =0, *N*/2, son comunes a ambos y tienen valor real.
6. El mayor componente de frecuencia (esto es, n = *N*/2*)* es $(2T)^{-1}$ Hz; puede aumentarse disminuyendo el espaciamiento entre muestras temporales.
7. El espaciamiento entre componentes de frecuencia es $(NT)^{-1}$ Hz; puede reducirse agregando ceros de aumento a la secuencia de muestras.
8. La relación exacta de los valores de la TRF depende de los factores constantes asignados en el algoritmo; un procedimiento bastante común es dividir por *N* de forma que los valores calculados son 1/*N* veces la TFD.

Algoritmo:

Aunque existen varias posibilidades, desarrollamos aquí un algoritmo de FFT del tipo Cooley-Tukey (algoritmo de decimación en frecuencia) que deberá implementarse como un programa MatLab, y compararse (en tiempo de CPU consumido) con la expresión original de la DFT.

El algoritmo FFT a implementar requiere una señal con un número de muestras *N* potencia de 2, y se resume en la parte (a) del siguiente gráfico (*N*=8):



El algoritmo se estructura en $\gamma = \log_2 N$ niveles de cómputo (l = 1, ..., γ). Para el desarrollo de este algoritmo deben tenerse en cuenta los siguientes puntos:

- Cada par de puntos de un cierto nivel se obtienen a partir de otros dos del nivel anterior. Estos dos puntos origen se dice que son *nodos duale*s.
- El espaciado entre dos nodos duales es $N/2^l$.
- Cómputo de nodos duales: mariposas. El cómputo de dos nuevos nodos a partir de dos nodos duales se denomina *mariposa*. Su estructura se muestra en la gráfica anterior (parte (b)), y se corresponde con la siguiente expresión:

$$x_l(k) = x_{l-1}(k) + W^P x_{l-1}\left(k + N/2^l\right)$$
$$x_l\left(k + N/2^l\right) = x_{l-1}(k) - W^P x_{l-1}\left(k + N/2^l\right)$$

  lo cual se puede simplificar a la estructura mostrada en la parte (c).

- Salto: una vez recorridos $N/2^l$ nodos consecutivos, se saltan los $N/2^l$ siguientes (ya que son los duales de los anteriores), y se prosigue el cómputo de mariposas.
- Determinación de W^P:

1. Escribir *k* (nodo actual) en forma de número binario de γ bits.
2. Desplazar γ - / bits a la derecha (rellenando con ceros).
3. Invertir el orden de los bits (p. ej., 100 pasa a 001). El resultado es *p*.

- En el último nivel se obtiene $\mathbf{x}_\gamma = \overline{X} \cdot \overline{X}$ es la DFT de la señal con sus muestras permutadas mediante inversión de bits. Por ejemplo:

$$\overline{X}(3) = \overline{X}(011) = \overline{X}(110) = X(6)$$

Materiales y equipos

1 Computadora con sistema operativo Windows y el programa Fourier y MATLAB y tarjeta de sonido.

Procedimiento

PARTE I. Algoritmo de la Transformada Rápida de Fourier (FFT) escrito en MATLAB.

1. Encienda la computadora y corra el programa MATLAB.
2. En MATLAB seleccione del menú **File** la opción **New** y en ella seleccione **M-file**. Aparecerá el Editor Depurador de archivos M.
   Escriba la siguiente función:

```
function x=mitrf(x)
% MITRF     Transformada Discreta de Fourier
% TRF(X) es la transformada discreta de Fourier del vector X. La
% longitud del vector X debe ser una potencia de 2.
%
% Vea también FFT, IFFT, FFT2, IFFT2, FFTSHIFT

n=size(x,2);
if (n-log(n)/log(2)) - fix(n-log(n)/log(2))~=0
  fprintf('longitud del vector = %.0f\n',n)
  error('La longitud del vector debe ser una potencia de 2.')
end

% Reordena el vector de datos complejos por la regla del bit inverso
j=n/2+1;
for i=2:n
  if i<j
    temp=x(i);
    x(i)=x(j);
    x(j)=temp;
  end
  k=n/2;
  while k<j & k~=0
    j=j-k;
    k=fix(k/2);
  end
  j=j+k;
end

% Realiza los cálculos del patrón de mariposa
j=sqrt(-1);
d=1;
while d<n
  e=d;
  d=d*2;
  u=1;
  w=exp(-pi/e*j)
  for b=1:e
```

```
    a=b;
    while a<=n
      f=a+e;
      temp=x(f)*u;
      x(f)=x(a)-temp;
      x(a)=x(a)+temp;
      a=a+d;
    end
    temp=real(u)+j*imag(temp);
    u=u*w;
  end
end
```

Guarde el archivo con el nombre MITRF.M, igual que el de la función

3. Compare la función de la transformada rápida de Fourier que usted hizo con el que ya viene incluida en MATLAB.
   Encuentre la TFD de las secuencias siguientes. (a) [1,0,1,0] (b) [1,1,0,0].
   Solución:
   En MATLAB escriba la siguiente secuencia:

```
» a = [1,0,1,0];
» Fd1 = fft(a)
» Fd2 = mitrf(a)
» b = [1,1,0,0];
» Fd3 = fft(b)
» Fd4 = mitrf(b)
```

   Son iguales los resultados de Fd1 y Fd2:_____.

   Son iguales los resultados de Fd3 y Fd4:_____.

4. Cuando la longitud del vector no es una potencia de 2 se puede usar el cálculo de la transformada discreta de Fourier por su definición.

```
function b = dftmtx(n)
%DFTMTX matriz de ls transformada Discreta de Fourier.
% DFTMTX(N) es la matriz compleja de N-por-N de valores alrededor
% del circulo-unitario cuyo producto punto con un vector columna
% de longitud N genera la transformada de Fourier discreta del
% vector.  DFTMTX(LENGTH(X))*X es igual que FFT(X).
%
% La matris de la transformada discreta inversa es
% CONJ(DFTMTX(N))/N.                Vea también FFT y IFFT.

f = 2*pi/n;             %Incremento angular.
w = (0:f:2*pi-f/2).' * sqrt(-1);          %Columna.
x = 0:n-1;              % Fila.
b = exp(-w*x);          % Exponenciación de producto punto.
```

5. Repita el numeral 3 del procedimiento para probar la función dftmtx anterior
   En MATLAB escriba la siguiente secuencia:

```
» a = a';     % debe ser un vector columna
» Fd1 = fft(a)
» Fd2 = dftmtx(length(a))*a
» b=b';
```

```
» Fd3 = fft(b)
» Fd4 = dftmtx(length(b))*b
```

Son iguales los resultados de Fd1 y Fd2:_____.

Son iguales los resultados de Fd3 y Fd4:_____.

PARTE II. Transformada DFT inversa (IDFT).

6. Esta transformada se expresa como:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{j2\pi nk/N}$$

Dada la gran cantidad de algoritmos y hardware específico existente para calcular la DFT directa , es conveniente encontrar la forma de obtener la IDFT a partir de la DFT. A continuación se implementarán tres métodos para realizar esto.

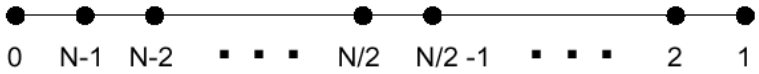
7. IDFT mediante rotaciones circulares:

Podemos observar que la diferencia fundamental entre la DTFT y su inversa está en el cambio de

signo de las exponenciales complejas. Por tanto, bastará realizar un cambio de variable $\omega \rightarrow -\omega$ para obtener la transformada inversa a partir de la directa:

$$x(n) = DTFT[X(-\omega)]$$

Para el caso de la DFT, las frecuencias positivas corresponden a k = 0,1, ...,(N/2) -1, y las negativas a k = (N/2), ..., N -1. Por tanto, la reordenación a realizar sobre la DFT será la siguiente:



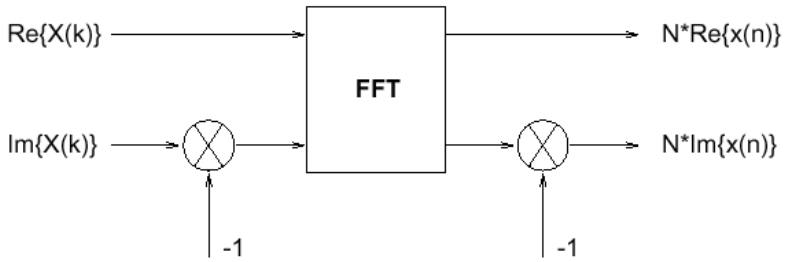
Esta reordenación puede realizarse como:

```
Y = [X(1) X(N:-1:2)];
```

Escribir una función MatLab que calcule la IDFT como:

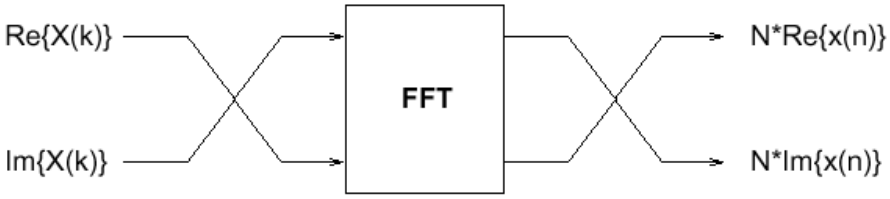
$$x(n) = \frac{1}{N} DFT[Y(k)]$$

8. IDFT mediante conjugados:
Este método corresponde al siguiente esquema:



Demostrar que el procesado anterior realiza una IDFT y escribir una función MatLab que lo implemente.

9. Repetir el apartado anterior para el siguiente esquema:



PARTE III. Transformada Rápida de Fourier de una señal de voz.

10. Capture una señal de voz de 32768 muestras con una frecuencia de muestreo de 8000 Hz, usando la tarjeta de sonido.
11. Presente la magnitud de la señal contra la frecuencia expresada en Hz.
12. Compruebe que la mayor potencia de la voz está en el rango de 300Hz a 3400Hz.
13. Salga del programa y apague la computadora.

Análisis de resultados

1. Encuentre el diagrama de flujo o flujograma del algoritmo de la transformada rápida de Fourier.
2. ¿Cuál es la razón de agregar ceros a la secuencia de números en el algoritmo FFT?
3. ¿Cuál es la utilidad de emplear la transformada de rápida de Fourier?
4. Usando la transformada rápida de Fourier demuestre la propiedad de modulación con un ejemplo.
5. Presente todas sus observaciones y conclusiones.

Investigación complementaria

1. Documentar la forma de obtener la transformada rápida de Fourier de una señal continúa utilizando SCILAB, además presente el archivo ejecutable.

Bibliografía

- Sistemas de Comunicaciones. Stremler, Ferrel G., Alfaomega.
- The MathWorks Inc. Manual del usuario de MATLAB
- C/C++ Interactive Reference Guide.